REMARKS

Applicant thanks the Examiner for clarifying the Examiner's positions and for providing a detailed explanation of the Examiner's Response to Applicant's traversal positions.

Entry of Preliminary Amendment filed April 15, 2005

As a preliminary matter, Applicant notes that a Preliminary Amendment was filed on April 15, 2005. The USPTO PAIR system (copies attached herewith) indicates that 40 claims (i.e., claims 1-4 and 5-41) were counted following the April 15, 2005 Preliminary Amendment. However, the present Office Action mailed April 27, 2005 does not include claim 41, which was added by the April 15, 2005 Preliminary Amendment.

Thus, it appears that the Preliminary Amendment, including new claim 41, may not have reached the Examiner prior to the preparation of the present Office Action, since the Examiner did not mention this claim or the Preliminary Amendment in the present Office Action. However, new claim 41 appears to have been counted (for fee purposes) and entered into the file.

Therefore, Applicant includes claim 41 in the present Amendment with the status identifier "Previously presented" and reiterates the request for entry of the same. Please note that Applicant has not resubmitted the excess claim fee payment letter, since the excess claim fee letter which accompanied the April 15, 2005 Preliminary Amendment appears to have been processed by the USPTO following the April 15, 2005. The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Thus, Claims 1-3 and 5-41 are all the claims presently pending in the application.

While Applicant believes that the claims are patentable over the cited references, either alone or in combination, to speed prosecution, Applicant amends independent claims 1 and 27 to

define more clearly and particularly the features of the claimed invention. Claim 8 also is amended merely to change its dependency from claim 1 to claim 2.

As mentioned above, Claim 41 (which was added by the Preliminary Amendment filed on April 15, 2005) is resubmitted herewith. Applicant reiterates that, while all of the claims are believed to be patentable over the cited references, either alone or in combination, to speed prosecution and allowance of the application, Applicant added claim 41 to define more clearly and particularly the features of the claimed invention. Accordingly, Applicant requests entry and consideration of claim 41, which was added by the Preliminary Amendment filed on April 15, 2005 and amended herewith to further clarify the language of the claim and the exemplary features of the present invention.

It is noted that the claim amendments are made only for more particularly pointing out the invention, and not for distinguishing the invention over the prior art, narrowing the claims or for any statutory requirements of patentability. Further, Applicants specifically state that no amendment to any claim herein should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

Claims 1-4, 5-26, and 28-40 stand rejected under 35 U.S.C. § 103(a) as being obvious over Fulton, III et al. (U.S. Patent No. 5,715,386; hereinafter "Fulton") in view of Garg ("A Methodology for Detection and Estimation of Software Aging", published November 1998. Claim 27 stands rejected under 35 U.S.C. § 103(a) as being obvious over Fulton in view of Garg, and further in view of Murphy, et al. (U.S. Publication No. 2003-015084 A1, filed January 10, 2000).

These rejections are respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

In an illustrative, non-limiting aspect of the present application, as defined by independent claim 1, a method of reducing a time for a computer system to recover from a degradation of performance in a hardware or a software in at least one first node of the computer system, includes monitoring a state of the at least one first node, and based on the monitoring, transferring a state of the at least one first node to a second node prior to the degradation in performance of the hardware or the software of the at least one first node. The method further includes proactively invoking a state migration functionality to reduce the recovery time and selectively running an application on the second node while the at least one first node is still operational. The application running on the second node corresponds to an application running on the hardware or the software on the at least one first node which has been determined by the monitoring as including the degradation in performance.

In another exemplary aspect of the invention, as defined by independent claim 19, a method of reducing a lack of performance in a computer system having at least one primary node and a secondary node, includes determining whether a failure or lack of performance is imminent, based on said determining, commanding a secondary node to start an application if it is not already running, and to begin reading a state and redo log from a memory coupled to said primary node and said secondary node, commanding the secondary node to apply the redo log to its state, commanding the primary node to begin mirroring its dynamic state updates to the secondary node as they occur, such that the secondary node's state is brought completely up to date with said primary node, judging whether the primary node has failed, and based on said judging, making the secondary node become the primary node.

Independent claims 26-28 and 34-40 recite somewhat similar features as independent claims 1 and 19.

In conventional devices, when a computer system suffers an unplanned failure, a certain amount of time is required to recover from the failure. This outage duration is undesirable.

For example, if the computer is a single node, stand-alone computer system, it must reboot and restart its application. If the computer is part of a multi-node high availability cluster architecture, it must failover (i.e., transfer) the application to another node in the cluster. During this recovery time, after either rebooting or failing-over the application to another node in a cluster environment, the recovering system must reload a stale copy of its state from disk, load a transaction redo log from disk, and attempt to reconstruct an up-to-date copy of that state by replaying that transaction redo log against the stale state (e.g., see specification at page 2, lines 2-11).

The claimed invention, on the other hand, provides an exemplary method (and system) for proactively reducing the outage duration by using the predicted outages to proactively trigger and manage existing failure recovery functionality (e.g., see specification at page 1, lines 13-16).

Particularly, the claimed invention exploits the ability to predict software outages or hardware failures to proactively migrate the state needed to quickly recover from an imminent outage onto another computer system (e.g., such as another node in a cluster environment) or a persistent storage medium (e.g., such as a hard disk) (e.g., see specification at page 5, lines 10-18). According to the present invention, in an exemplary, non-limiting aspect, the system that was about to fail (e.g., the "failing computer") could "prime" another computer (e.g., the "failover target") by directing it to read from disk the stale state and the redo log, and then direct the failover target computer to begin applying the redo log to the stale state. Moreover, upon

discovery of its imminent demise, the failing computer could mirror all transactions to the failover target, bringing the failover target's state even more up-to-date. When the primary computer does finally fail, the failover target would have an up-to-date copy of the state, the lengthy reloading of state and redo log from disk would be avoided, the outage would be shortened, and the system availability would be improved (e.g., see specification at page 6, lines 7-12).

Thus, the present invention reduces outage duration by using the ability to predict outages to proactively trigger and execute functionality whose effect is to reduce that outage's duration. Further, in the case of a single node, the amount of time required to reconstruct the node's state after the outage has occurred can be reduced significantly (e.g., see specification at page 6, lines 7-12).

II. THE PRIOR ART REJECTIONS

A. Claims 1-4, 5-26, and 28-40 stand rejected under 35 U.S.C. § 103(a) as being obvious from Fulton in view of Garg.

Applicant incorporates herein by reference in their entirety the traversal arguments set forth in the Amendment under 37 C.F.R. § 1.111 filed on September 30, 2004, the Amendment under 37 C.F.R. § 1.116 filed on February 14, 2005, and the Preliminary Amendment under 37 C.F.R. § 1.114 filed on April 15, 2005, for the Examiner's convenience.

As mentioned above, Applicant thanks the Examiner for clarifying the Examiner's positions and for providing a detailed explanation of the Examiner's Response to Applicant's traversal positions.

However, for the following reasons, Applicant respectfully traverses these rejections.

First, while Applicant believes that claim 1 is patentable over the cited references, either alone or in combination, to speed prosecution, Applicant amends claim 1 to define more clearly and particularly the features of the claimed invention. Claim 8 also is amended merely to change its dependency from claim 1 to claim 2.

As mentioned above, in conventional devices, such as Fulton, when a computer system suffers an unplanned failure, a certain amount of time is required to recover from the failure. This outage duration is undesirable.

For example, if the computer is a single node, stand-alone computer system, it must reboot and restart its application. If the computer is part of a multi-node high availability cluster architecture, it must failover (i.e., transfer) the application to another node in the cluster. During this recovery time, after either rebooting or failing-over the application to another node in a cluster environment, the recovering system must reload a stale copy of its state from disk, load a transaction redo log from disk, and attempt to reconstruct an up-to-date copy of that state by replaying that transaction redo log against the stale state (e.g., see specification at page 2, lines 2-11).

The Examiner relies on Fulton for disclosing allegedly similar features (e.g., see Office Action at page 7, lines 6-9, citing Fulton at column 9, lines 11-18, and column 10, lines 9-15). However, Applicant respectfully submits that Fulton does not disclose or suggest these features of the claimed invention, for which Fulton is relied upon.

For example, Fulton discloses that when daimon 104 determines that a process has crashed, it restarts the process by restoring any critical memory 115 from a critical memory copy 125, and if a file log 127 exists, consuming the messages in the log file 127 (e.g., see Fulton at column 9, lines 10-18). Indeed, Fulton discloses that copies of the state of the process are stored

in files and include any copies of critical memory 125 and any log file 127 for the process (e.g., see Fulton at column 10, lines 9-15).

Thus, like the conventional methods and systems described by Applicant in the present application, after a failure occurs, Fulton must reload a stale copy of its state from disk or another node, load a transaction redo log from disk or another node, and attempt to reconstruct an up-to-date copy of that state by replaying that transaction redo log against the stale state (e.g., see specification at page 2, lines 2-11). Again, all of these processes occur after the first node has failed.

Turning to the language of the claims, independent claim 1 recites a method of reducing a time for a computer system to recover from a degradation of performance in a hardware or a software in at least one first node of the computer system, including:

> *monitoring a state of said at least one first node;*
>
> *based on said monitoring, transferring a state of said at least one first node to a second node prior to said degradation in performance of said hardware or said software of said at least one first node;*
>
> *proactively invoking a state migration functionality to reduce said recovery time; and*
>
> *selectively running an application on said second node while the at least one first node is still operational,*
>
> *wherein said application running on said second node corresponds to an application running on said hardware or said software on said at least one first node which has been determined by said monitoring as including said degradation in performance* (emphasis added).

Thus, the claimed invention provides an exemplary method (and system) for proactively reducing the outage duration by using the predicted outages to proactively trigger and manage

existing failure recovery functionality (e.g., see specification at page 1, lines 13-16). That is, the claimed invention "primes" another computer (e.g., the "failover target") by directing it to read from disk the stale state and the redo log, and then direct the failover target computer to begin applying the redo log to the stale state, while the first node is still operational.

The claimed invention exploits the ability to predict software outages or hardware failures to proactively migrate the state needed to quickly recover from an imminent outage onto another computer system (e.g., such as another node in a cluster environment) or a persistent storage medium (e.g., such as a hard disk) (e.g., see specification at page 5, lines 10-18).

For example, according to the claimed invention, the system that was about to fail (e.g., the "failing computer") could "prime" another computer (e.g., the "failover target") by directing it to read from disk the stale state and the redo log, and then direct the failover target computer to begin applying the redo log to the stale state. Moreover, upon discovery of its imminent demise, the failing computer could mirror all transactions to the failover target, bringing the failover target's state even more up-to-date.

Thus, when the primary computer does finally fail, the failover target would have an up-to-date copy of the state, the lengthy reloading of state and redo log from disk would be avoided, the outage would be shortened, and the system availability would be improved (e.g., see specification at page 6, lines 7-12). Accordingly, the present invention reduces outage duration (e.g., see specification at page 6, lines 7-12).

As mentioned above, Fulton and Garg, either individually or in combination, do not disclose or suggest at least "*selectively running an application on said second node while the at least one first node is still operational, wherein said application running on said second node corresponds to an application running on said hardware or said software on said at least one*

first node which has been determined by said monitoring as including said degradation in performance" (emphasis added), as recited in independent claim 1.

Instead, Fulton merely discloses that when daimon 104 determines that a process has crashed (i.e., after it has crashed), it restarts the process by restoring any critical memory 115 from a critical memory copy 125 and a file log 127 (e.g., see Fulton at column 9, lines 10-18; column 10, lines 9-15), or if a failure does not occur, performing the rejuvenation during a most idle time takes less time than performing rejuvenation during a non-idle time (see Fulton at column 5, lines 55-60).

Thus, like the conventional methods and systems described by Applicant in the present application, after a failure occurs, Fulton must reload a stale copy of its state from disk or another node, load a transaction redo log from disk or another node, and attempt to reconstruct an up-to-date copy of that state by replaying that transaction redo log against the stale state (e.g., see specification at page 2, lines 2-11). Again, all of these processes occur after the first node has failed.

On the other hand, Garg merely relates to a method of detecting and estimating aging in operational software. Garg monitors operating system resource usage and system activity and discloses using an "Estimated time to exhaustion" metric to compare the effect of aging on different system resources and also in the identification of important resources to monitor and manage.

In other words, Garg is concerned with identifying the time (i.e., the occasion) for performing the rejuvenation. That is, Garg is trying to predict when to perform the rejuvenation such that the operational software can be stopped and then restarted in a clean state (e.g., see Garg at page 1, column 2, first paragraph, lines 7-10; see also page 10, first paragraph).

As with Fulton, however, Garg does not disclose or suggest at least "*selectively running an application on said second node while the at least one first node is still operational, wherein said application running on said second node corresponds to an application running on said hardware or said software on said at least one first node which has been determined by said monitoring as including said degradation in performance*" (emphasis added), as recited in independent claim 1.

Indeed, like Fulton, Garg specifically discloses that "*[w]hen a monitored machine fails, it is restarted/rebooted and the agent process is also restarted*" (see Garg at page 4, first column, first paragraph, line 11-13). That is, after the monitored machine fails, then Garg restarts/reboots the machine.

Thus, when considered as a whole for what they fairly teach to the ordinarily skilled artisan, Fulton and Garg, either individually or in combination, clearly do not disclose or suggest all of the features of the claimed invention, as recited in independent claim 1. Therefore, the Examiner is requested to reconsider and withdraw the rejection of independent claim 1 (and dependent claims 2, 3, and 5-18).

On the other hand, independent claim 19 recites, *inter alia*, a method of reducing a lack of performance in a computer system having at least one primary node and a secondary node, including:

> *determining whether a failure or lack of performance is imminent;*
>
> *based on said determining, commanding a secondary node to start an application if it is not already running, and to begin reading a state and redo log from a memory coupled to said primary node and said secondary node;*

commanding the secondary node to apply the redo log to its state;

commanding the primary node to begin ***mirroring*** *its dynamic state updates to the secondary node as they occur, such that the secondary node's state is brought completely up to date with said primary node;*

judging whether the primary node has failed; and

based on said judging, making the secondary node become the primary node (emphasis added).

Thus, according to the novel and unobvious features of the present invention, a system that is about to fail (e.g., the first node, or the "failing computer", etc.) can "prime" another computer (e.g., the secondary node, or the "failover target", etc.) by directing it to read from disk the stale state and the redo log, and then direct the failover target computer to begin applying the redo log to the stale state (e.g., see specification at page 5, lines 19-23).

Moreover, upon discovery of its imminent demise, the failing computer could **mirror** all transactions to the failover target, bringing the failover target's state even more up-to-date. When the primary computer does finally fail, the failover target would have an up-to-date copy of the state, the lengthy reloading of state and redo log from disk would be avoided, the outage would be shortened, and the system availability would be improved (e.g., see specification at page 6, lines 7-12).

For somewhat similar reasons as those set forth above with respect to claim 1, when considered as a whole for what they fairly teach to the ordinarily skilled artisan, Fulton and Garg, either individually or in combination, clearly do not disclose or suggest all of the features of the claimed invention, as recited in independent claim 19.

That is, neither Fulton nor Garg discloses or suggests at least "*commanding a secondary node to start an application if it is not already running, and to begin reading a state and redo log from a memory coupled to said primary node and said secondary node; commanding the secondary node to apply the redo log to its state; commanding the primary node to begin mirroring its dynamic state updates to the secondary node as they occur, such that the secondary node's state is brought completely up to date with said primary node*" (emphasis added), as recited in independent claim 19.

Instead, both Fulton and Garg wait for the process to fail, and then restart/reboot the machine based on stored data.

Therefore, the Examiner is requested to reconsider and withdraw the rejection of independent claim 19 (and dependent claims 20-25).

Applicants submit that independent claims 26-28 and 34-40 are patentable over Fulton and Garg for somewhat similar reasons as those set forth above, as well as for the reasons set forth in the Amendment under 37 C.F.R. § 1.111 filed on September 30, 2004, the Amendment under 37 C.F.R. § 1.116 filed on February 14, 2005, and the Preliminary Amendment under 37 C.F.R. § 1.114 filed on April 15, 2005, which are incorporated herein by reference in their entirety, for the Examiner's convenience.

For example, independent claim 28 recites a method of reducing a degradation of performance in a computer system having at least one primary node and a secondary node, comprising:

> *determining whether a degradation of performance of said primary node is imminent;*
>
> *based on said determining, commanding said secondary node to start an application if it is not already running;*

replicating, by said secondary node, a state of said primary node;

and

passing control to said secondary node from said primary node

(emphasis added).

Applicants respectfully submit that Fulton and Garg, either alone or in combination, do not disclose or suggest all of the novel and unobvious features of independent claims 26-28 and 34-40.

Accordingly, the Examiner respectfully is requested to withdraw the rejection of claims 1-3, 5-26, and 28-40 and permit these claims to pass to immediate allowance.

B. Claim 27 stands rejected under 35 U.S.C. § 103(a) as being obvious over Fulton in view of Garg, and further in view of Murphy.

First, while Applicant believes that claim 27 is patentable over the cited references, either alone or in combination, to speed prosecution, Applicant amends claim 27 to define more clearly and particularly the features of the claimed invention.

For somewhat similar reasons as those set forth above, Applicants respectfully reiterate that it clearly would not have been obvious to combine Fulton and Garg to arrive at the claimed invention defined by claim 27.

For example, independent claim 27 recites, *inter alia*, a method of reducing a degradation period of a Web hosting machine, including:

monitoring a performance of said Web hosting machine;

transferring a state of said Web hosting machine to a second machine when a degradation of said performance occurs in said Web hosting machine; and

selectively running an application on said second machine based on said state while the Web hosting machine is still operational,

wherein said application running on said second machine corresponds to an application running on said Web hosting machine which has been determined by said monitoring as including said degradation in performance (emphasis added).

Moreover, Applicant reiterates that Murphy would not have made up for the deficiencies of Fulton and Garg. Indeed, Murphy is not even relied upon for the features of reducing a degradation period or reducing a time to recover from a degradation of performance in a hardware or a software, as claimed. Instead, Murphy is relied upon for showing "*a node for a Web hosting machine*" (see Office Action at page 23, lines 7-8).

Thus, Applicants submit that Fulton, Garg, and Murphy, either alone or in combination, do not disclose or suggest all of the novel and unobvious features of the claimed invention.

Therefore, the Examiner is requested to reconsider and withdraw this rejection and permit claim 27 to pass to immediate allowance.

III. CLAIM 41:

Turning to claim 41, claim 41 recites a method of reducing a time for a computer system to recover from a degradation of performance in a hardware or a software in at least one first node of said computer system, including:

monitoring a state of said at least one first node;

predicting an outage of said hardware or said software based on monitoring;

based on said monitoring, transferring a state of said at least one first node to a second node prior to said degradation in performance of said hardware or said software of said at least one first node;

proactively invoking a state migration functionality to reduce said recovery time, wherein said proactively invoking includes migrating a dynamic state to stable storage of said second node, said second node being accessible to a recovering agent, to reduce an amount of time required by said recovering agent; and

connecting said at least one first node and said second node to a shared memory containing a stale state of the at least one first node and a redo log,

wherein said shared memory includes at least one of a shared storage medium, a shared storage disk and a shared network,

wherein said degradation of performance comprises one of an outage and a failure,

wherein said second node selectively includes an application running corresponding to an application failing on said at least one first node while the at least one first node is still operational,

wherein said state transfer from said at least one first node to said second node occurs while the at least one first node is still operational, and

wherein said predicting comprises providing a failure predictor on at least one of said at least one first node and said second node, for commanding the at least one first node to start an application if not already running while the at least one first node is still operational, and commanding the second node to begin reading a state of said at least one node and redo log from the shared memory while the at least one first node is still operational (emphasis added).

For somewhat similar reasons as those set forth above, Applicants respectfully reiterate that it clearly would not have been obvious to combine Fulton and Garg to arrive at the claimed invention defined by claim 41.

IV. CONCLUSION

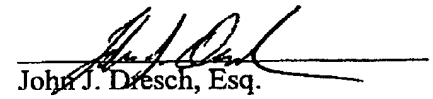
In view of the foregoing, Applicants submit that claims 1-3 and 5-41, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: July 27, 2005



John J. Dresch, Esq.
Registration No. 46,672
Sean M. McGinn, Esq.
Registration No. 34,386

McGinn & Gibb, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254